# Asset-centric security-aware service selection

Tziakouris, Giannis; Zinonos, Marios; Chothia, Tom; Bahsoon, Rami

*DOI:*
10.1109/BigDataCongress.2016.50

# Asset-Centric Security-Aware Service Selection

Giannis Tziakouris, Marios Zinonos, Tom Chothia, Rami Bahsoon
School of Computer Science, University of Birmingham
Birmingham, United Kingdom
{yxt101, mxz178, t.chothia, r.bahsoon}@cs.bham.ac.uk

***Abstract*—Catering for the runtime security of users and their assets (e.g. files, accounts, etc.) in service oriented environments is a challenging problem. We motivate the need for an adaptive framework that selects online services according to the runtime security requirements and cost constraints of assets. We report on a market-inspired approach (i.e. reversed Posted-Offer auction) that satisfies multiple, heterogeneous requests for online services in environments with shared and scarce resources. The solution is tested on the specific area of Cloud storage services.**

**Keywords-security; service selection; market-inspired methodology; asset-centric;**

## I. Introduction

We perceive service repositories as marketplaces consisting of diverse, on-demand resources and services that can be traded to users for satisfying their functional and security requirements. The service markets and repositories have been witnessing significant increase in the number of services offering comparable capabilities but tend to vary in their security provision. Despite their similarities, each service is associated with different security features, costs, strengths and limitations from which users are inquired to select the one that best satisfy their needs. Their selection can be a challenging, time-consuming and expensive task, as users need to conduct a thorough survey on the documentation of all available services and service providers (SPs). The screening can inform an optimal selection decision. It can also inform the extent to which the selection can protect the users' assets. By users' assets, we refer to contextual (e.g. location) and contentual (e.g. files) data maintained by users. The problem is complex as we operate in elastic environments with multiple self-interested users, each maintaining various assets that are associated with different security requirements and cost constraints.

In spite of the vast research conducted on service selection, only a small fraction of the work is concerned with the security of services as either an additional quality dimension or as an explicit one. However, these systems appear to be incapable of handling diverse user requests that necessitate the enforcement of different security policies for different assets and the trade-offs that might arise from them. This limitation outcomes into the provision of "one service for all", which leads to higher costs (users pay for services and resources that they do not utilize) and unmet security requirements (in terms of individual assets). Furthermore, existing service selection frameworks tend to focus on the satisfaction of the security goals as an "aggregate quality". They tend to provide limited or no support for the composite security attributes, their global satisfaction and trade-offs. Added, the Ad-hoc allocation mechanisms utilized by existing service selection systems can cause resource starvation when applied in environments with scarce and shared resources. This is due to their simplistic "first come first served" allocation policy, which does not allow the prioritization of user requests based on their security requirements and criticality. This can result in the wasteful allocation of services/resources to users that do not wish their immediate utilization, while refusing provision to users that they need it. A solution to the problem is to engineer security-aware service selection frameworks that leverage market-inspired allocation methodologies. The choice of market-inspired solution is justified for its proven ability in converging towards "optimal" allocation in scenarios involving multiple users competing for shared resources in dynamic and elastic environments. Our market-inspired solution aims to select service(s) that can best satisfy competing users' requirements for both security and price of provision. Our work features the following novelties:

- Develop a security-aware service selection framework that utilizes market-inspired allocation methodologies and the Weighted Linear Combination algorithm [1].
- Demonstrate the applicability of our framework on the specific-area of Cloud storage services. We suggest a list of security attributes that can drive the selection of services in the context of Cloud storage services. We also show how the selection can be sensitive to these attributes.

The remaining of the paper is structured as follows. Section II reviews related work. Section III introduces the proposed framework. Section IV applies our framework on Cloud storage services. Section V tests and evaluates our approach. Section VI concludes with future work.

## II. Related Work

The services computing literature is saturated with contributions that relate to dynamic service selection. We only review closely related work that is concerned with security.

The work of [2] proposes a service selection framework that identifies web services based on user search keyword criteria, while preserving the privacy of users and SPs. The proposed system conceals the search criteria of users and the provision rules of SPs via the utilization of the Private Matching (PM) protocol. PM derives the intersection between the inputs of a user and a SP without leaking any additional information.

Mouratidis et al. [3], proposes a framework that selects Cloud services and integrates them in software products based on the security requirements of clients. When a client needs to select a Cloud service, it constructs a risk assessment document describing possible threats. Based on the document, the framework models security/privacy requirements. Following, each modeled requirement is assigned weights to reflect their significance for the client. Finally, the weights of each service are summed up, from which the one acquired the highest weight score is selected and integrated in the software.

Costante et al. [4], propose an approach for Web service selection and composition based on user privacy preferences. The proposed work utilizes AND/OR trees to represent the orchestration schema, component services and their privacy policies. Based on this representation, the system composes suitable Web services that satisfy a user's privacy preferences. To achieve this, the system ranks composite Web services with respect to their privacy level. The privacy level quantifies the risk of misuse of personal data based on three dimensions: sensitivity, visibility and retention period of information.

Khani et al. [5], suggest an intrusion-tolerant composite Web Service (WS) system, that performs selection of WSs based on their security vulnerabilities. The authors utilize penetration tools against WSs to identify security gabs. Based on the auditing findings and the user requirements, the framework selects the WS with the highest performance and resilience.

## III. Framework Design

This section reasons for the selection of a market-inspired architecture as the foundation of our framework. Additionally, it introduces and examines the entities and operation phases comprising our framework. The source code of our system can be found at: https://github.com/GiannisT/AssetsAwareSecurity.git.

### A. Market-Inspired Architecture

Following the argument of [6], we view service repositories as marketplaces supporting the "5th utility" and delivering on-demand services that are traded in a similar manner as traditional utilities. The services in the market serve the security goals that users require when operating in dynamic environments. As a result, assets can be secured by practicing service selection, based on their runtime security requirements.

Market inspired methodologies can be an effective optimization mechanism for the continuous satisfaction of the varying security requirements of multiple self-interested users and their assets. Market models promote transparency in the way services and resources are traded as their operations are based on systematic procedure; henceforth, promoting trust between SPs and users. Furthermore, the decentralized decision making nature of market methodologies promote the development of more dependable and elastic frameworks as it can eliminate the single point of failure and enable the isolation of users from the unsecure collaborations of SPs. In addition, market frameworks allow users to handle their own security requirements and data, which simplifies the concurrent management of multiple heterogeneous user requests as a major part of the computations and decisions are performed in a decentralized manner. This also mitigates trade-offs and conflicts that can rise between users. Lastly, the usage of such methodologies allow for the regulation of the supply and demand of services/resources at market equilibrium.

### B. Framework Entities

*1) Agents:* Agents are self-interested, decentralized, autonomous entities representing users. An agent is concerned with monitoring two events, the creation of assets and their runtime modification (e.g. the addition of data in a text file, etc.). When an asset is created, an agent requests the selection of a service that best satisfies its runtime security requirements and cost constraints. Contrary, if an asset is modified, the agent re-evaluates the current service to determine if it can still satisfy the existing security requirements and cost constraints. If the assessment illustrates that it is sufficient no action is required, whereas if its deemed insufficient, the agent inquires the selection of a new service. To trigger service selection, a user forms a *bid* which contains the security specification and cost constrains of an asset and forwards it to the auctioneer for auctioning with suitable services.

*2) Service Providers (SP):* SPs are companies that trade their services to agents for a fee. Each SP is responsible for submitting its offers (*asks*) to the auctioneer. Each ask encloses a description of the type of the service that is offering and the security features provided. If a match occurs, the winner SP receives payment from the agent and makes the requested service(s) available for usage.

*3) Auctioneer :* The market auctioneer is responsible for matching asks with bids. We assume that we operate in a market with a large number of agents, where the demand often surpasses the supply and consequently agents may need to compete for acquiring service. To perform auctions the auctioneer utilizes a reversed version of the Posted-Offer model [7]. Our reversed Posted-Offer model is founded on a take-it-or-leave-it basis. The auctioneer collects all available bids for a certain trading period and it is up to the SPs to accept or decline the requests. The SPs serve the agents with the highest bidding prices, until they are not able to cater for more agents due to resource exhaustion.

### C. Framework Operation Phases

Our system consists of four phases: Asset Monitoring, Requirements Assemble, Discovery and Feedback.

*1) Monitor Phase:* Sensors are deployed at the agent side for monitoring the creation and runtime modification of assets. When such an event occurs, the requirements assemble phase is triggered (Fig. 1, step 1).

*2) Requirements Assemble Phase:* In case where an asset is created, an agent must either form a new bid or select an existing one (older bid) to specify the security features required for his/her asset along with the significance of each feature for the agent (Fig. 1, step 2). The significance of each feature is denoted with a number between 1-3, where 1 is low, 2 is medium and 3 is high. Following, the agent specifies the cost that is willing to pay to obtain the needed service(s). Next, the

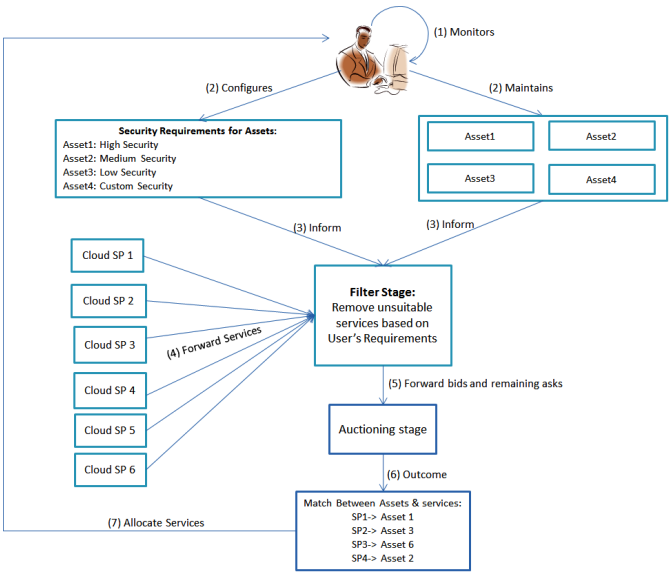


Figure 1. Shows the steps performed by our framework to select a service

agent chooses between two service selection methods, namely: the explicit selection and implicit selection. In the explicit selection, the auctioneer is only allowed to select service(s) that satisfy the exact security requirements specified in a bid (perfect match). In case that a bid cannot be satisfied (e.g. additional non requested security features, missing security features, etc.) a match cannot occur. Similarly, the implicit selection methodology aims to select service(s) that can satisfy the exact requirements set in a bid, however the auctioneer is not restricted to a perfect solution. The auctioneer is able to select the service with the least degree of variation from the specification of an agent. To make the proposed framework more convenient, we offer a set of three preloaded policies which allow agents to quickly select between low, medium and high level of security provision for their assets. In case where an agent requires to form its own security policy to tailor its security provision, the custom security policy can be selected.

Contrary, when an asset is modified, the agent must re-evaluate the selected service(s) to determine if they still meet the requirements of the asset. If the service is found adequate no action is performed, whereas if it is insufficient a bid is formed to inquire the selection of new service(s). Finally, in both cases the bid is forwarded to the auctioneer for auctioning with available asks (Fig. 1, step 3).

*3) Discovery Phase:* Once the auctioneer receives a bid $b$ from an agent the discovery phase is initiated. The auctioneer retrieves all asks (denoted as $Os_1, s_2, s_3, s_4 \dots s_n$) from the SPs (denoted as $\{SP_1, SP_2, SP_3, \dots, SP_n\}$) that an agent is registered with (Fig. 1, step 4). Following, the auctioneer initiates the preliminary elimination of unsuitable asks that violate the functional requirements of an agent's bid. The functional specification of every ask in the set $O\{s_1, s_2, s_3, s_4 \dots s_n\}$is compared with the functional requirements specified in an agent's bid. In case where the requirements (case specific) of an agent are met, then the examined service $s_i$ is added to the set of suitable services/asks $O'\{s_2, s_{3,} \dots, s_n\}$. Upon completion of the filtering stage, a utility score is calculated for each ask in the set $O'\{s_2, s_3 \dots s_n\}$ (Fig. 1, step 5) in accordance to bid $b$.

The utility score aims to quantify the level of security provision of each service and enable us to identify the dominant service(s). As the specification of bids and asks contain multiple attributes, the multi-attribute decision making algorithm: Weighted Linear Combination (WLC) is used for selecting suitable service. The WLC operates by adding contributions from each attribute to obtain a global score. In our case we assess each of the security attributes contained in a bid to examine if they are satisfied by their respective attributes in the asks. If an attribute is satisfied we set a score $h$ (we have chosen the value 10), where if the attribute is not satisfied we set a lower score $l$ (i.e 5). Lastly, we multiply the appointed scores for each of the examined attributes with their associated weights $w$ and then sum all of them to obtain the total utility score $Ut$ of each $SP_i$ (Equ. 1).

$$Ut = \sum_{i=1}^{n} w_i * (r_i || l_i) \tag{1}$$

Once the utility score of all asks is calculated, the set of services $O'\{s_2, s_{3,} \dots, s_n\}$ is ordered in a descending order based on their utility scores. In case that the agent has selected the implicit method for service selection, the first service in the ordered set $O'\{s_2, s_{3,} \dots, s_n\}$is selected as the optimal service, whereas if the explicit method is selected the first service is only selected if all of the attributes in a bid were satisfied by the attributes specified in an ask. In case of an imperfect match no service is selected, as the security requirements of the agent are violated (Fig. 1, step 6). Upon selection of a suitable service, the bid is forwarded to the matched SP. If the demand for the required service(s) exceeds the market supply, then the agent needs to engage in auctioning with rival agents to acquire the service. The auctioning is performed based on our reversed Posted-Offer auction model, in which after each auctioning round SPs need to either accept or decline available bids based on their bidding prices. Agents are sorted in a descending price order, where the first agents obtain service until a SP cannot cater for more agents.

*4) Feedback:* Upon service selection, the auctioneer notifies the winner SP and agent for the match and releases the service instance for allocation (Fig.1, step 7).

## IV. Application in Cloud Storage Services

This section promotes the effectiveness of our framework via its direct application on Cloud storage services. We introduce a list of security attributes (used by our prototype system) for describing and selecting Cloud storage services. We then present a scenario to illustrate the workings of our framework.

### A. Cloud Storage Security Descriptors

To capture the runtime security requirements and cost constraints of assets along with the specification of Cloud storage services, we have formed a list of fifteen attributes/security

descriptors that agents and SPs use to form their bids and asks respectively:

***Encryption At Rest:*** Illustrates whether an asset is encrypted in the machines of a Cloud storage provider.

***Encryption At Transit:*** Shows if the communication links between an agent and a Cloud provider are encrypted.

***Password Protected Files:*** Demonstrates if a file is protected by a separate password.

***File Versioning*:** Indicates if a SP keeps records of different versions of a file.

***Encryption Keys Concealed:*** Illustrates if a provider allow agents to use their own encryption keys to improve their privacy from the SP itself.

***Auto Synch:*** Illustrates if local files are automatically updated on Cloud storage.

***Secure Key Management:*** Shows if the encryption keys are stored securely by SPs.

***Credential Recovery:*** Designates if a user can obtain its forgotten username and password.

***Share Data:*** Shows if an agent wants to share data with third parties.

***Audit Logs:*** Illustrates if a storage provider records the operations of agents.

***Proxy Support:*** Shows if a SP allows the connection of proxies.

***Different Key Per File:*** Demonstrates if an agent requires a (dedicated) unique key for the encryption of a file.

***Permanent Deletion:*** Signifies if a storage provider permanently deletes files upon request or if they are instead archived on SP data-centers.

***Data-center Location****:* Demonstrates the hosted location of the data-centers of a Cloud storage provider.

***Certification:*** Illustrates the standards and regulations (e.g. ISO 27001) required for each asset.

***Cost:*** Reflects the amount of money that an agent is willing to pay for storing an asset.

Both agents and SPs need to specify their values for each of the above attributes. The values used by agents and SPs when forming their bids and asks are (except from the data-center location and certification): (i) "yes" which indicates that the certain feature is offered (in ask) or needed (bid), (ii) "no" which indicates that this feature is not offered/needed and (iii) "irrelevant" which demonstrates that an agent is not concerned with this attribute and consequently should not be considered when selecting services. The valuations used for the remaining attributes are: Data-center Location {"irrelevant", "Asia", "EU", "USA", "North America", "Russia", "Oceania"}; and Certification {"ISO 27001", "Safe Harbor", "FERPA", "HIPPA"}. Finally, agents need to specify the price that are willing to pay for acquiring services and the importance (weights) of each attribute (except cost) for them by assigning a number between 1(low) and 3 (high).

### *B. Cloud Storage Scenario*

For the purposes of this scenario we have considered the following Cloud storage providers: CloudMe [8], Cubby [9], DropBox [10], GoogleDrive [11], Mega [12], OneDrive [13], SpiderOak [14], YandexDisk [15] and BearDataShare [16]. For each of our selected storage services we have derived a security specification (ask) reflecting the security attributes previously described in this section. The derived security information for the selected providers is summarized in Table I. The extracted data is based on the free plans (except SpiderOak, $7/month) offered by the aforementioned providers in real life.

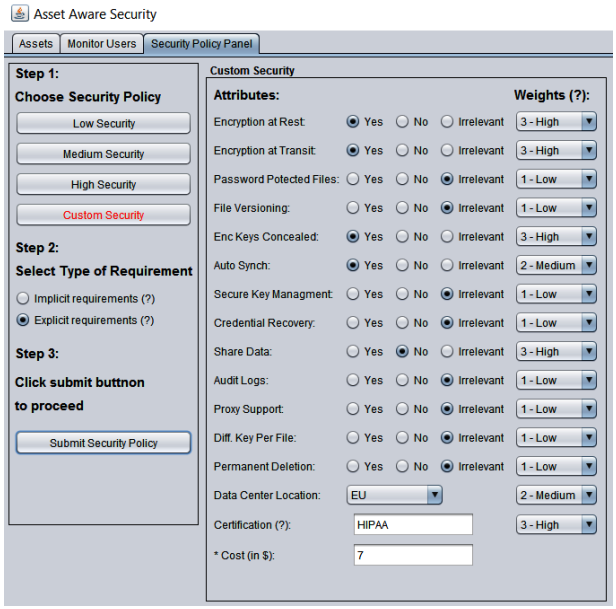


Figure 2. Illustrates the information enclosed in a bid

To illustrate the workings of our framework in the area of Cloud storage services, we consider the following scenario: We operate in a market utilizing the selected storage providers and 50 agents. Due to limited resources a number of agents are not be able to acquire resources. One of the agents, i.e. AgentX maintains a medical form that necessitate a Cloud storage service that utilizes low security, as it does not contain any sensitive medical data (not filled). To perform this, AgentX forms a bid to reflect the current requirements of his asset. The agent selects the preset, low security policy which inquires the selection of a storage that only provides: encryption at transit, file synchronization and credential recovery, where all remaining attributes are set to "no". The low security policy aims to discover a service that can host AgentX's file in a fairly secure environment, while warranting low costs and minimal resource overhead. Additionally, AgentX selects the implicit discovery mode as he is not concerned if his security requirements are exactly met. The bid is then forwarded to the auctioneer. Upon bid reception, the auctioneer eliminates unsuitable asks in terms of storage size (functional requirement). As the storage size offered by the considered services exceed the file size of AgentX's form, no services are eliminated. Thereafter, for each service, a security utility score is calculated (with the assistance of Equ. 1). The results illustrate that OneDrive meets all AgentX's requirements, thus acquiring the highest utility score of 160, followed by CloudMe, SpiderOak, GoogleDrive at 155, DropBox at 150,

| | *CloudMe* | *Cubby* | *DropBox* | *GoogleDrive* | *Mega* | *OneDrive* | *SpiderOak* | *YandexDisk* | *BearDataShare* |
|---|---|---|---|---|---|---|---|---|---|
| *Encryption At Rest* | yes-no | yes | yes | yes | yes | no | yes | no | yes |
| *Encryption At Transit* | yes | yes | yes | yes | yes | yes | yes | yes | yes |
| *Password Protected Files* | no | yes | no | no | no | yes | no | no | no |
| *File Versioning* | yes | yes | yes | yes | yes | no | yes | no | yes |
| *Encryption Keys Concealed* | yes | yes | no | no | yes | no | yes | no | no |
| *Auto Synch* | yes-no | yes-no | yes-no | yes-no | yes-no | yes-no | yes | yes-no | yes |
| *Secure Key Management* | no | yes | yes | yes | yes | yes | no | no | yes |
| *Credential Recovery* | yes | yes | yes | yes | no | yes | no | yes | no |
| *Share Data* | yes-no | yes-no | yes-no | yes-no | yes-no | yes-no | yes-no | yes-no | yes |
| *Audit Logs* | yes | yes | yes | yes | yes | yes | yes | yes | yes |
| *Proxy Support* | yes-no | yes-no | yes-no | yes-no | yes-no | no | yes-no | yes-no | yes |
| *Different Key Per File* | yes-no | yes-no | no | no | no | no | no | no | no |
| *Permanent Deletion* | yes | yes | yes* | yes | yes | no | no | yes | no |
| *Data-center Location* | EU | USA | USA | USA | Oceania | USA | EU | Russia | EU |
| *Certification* | HIPAA, Safe Harbor | HIPAA | ISO 27001, Safe Harbor | ISO 27001, FERPA, HIPAA, Safe Harbor | HIPAA | ISO 27001 | ISO 27001, FERPA, HIPAA | / | ISO 27001 |

* Indicates that a provider permanently deletes data but within some time.

Table I
SUMMARIZES THE SECURITY ATTRIBUTES (ASK) OF EACH CLOUD STORAGE PROVIDER

YandexDisk at 145, Cubby at 140 and BearDataShare and Mega at 130 points. For this scenario, we assume that the service supply of OneDrive exceeds the market demand, thus no auctioning is necessary for acquiring storage.

Now lets consider that AgentX fills its medical form with sensitive data and reforms his bid to reflect the new requirements at hand. AgentX selects the custom security policy to inquire tailored security provision. The new bid (Fig. 2) entail the following explicit requirements: Encryption at rest and transit; encryption keys concealed from the SP, auto-synch, no share data, the data must be stored within EU, HIPAA (Health Insurance Portability and Accountability Act) certification; and the cost that AgentX is willing to pay for storage is $7.

Once the medical form is modified the new bid is forwarded to the auctioneer. Upon reception, the bid is assessed to determine which service can better satisfy the new security requirements. Due to the selection of explicit requirements only two storage services (i.e. SpiderOak and CloudMe) can satisfy the exact security requirements of AgentX. Despite the discovery of two suitable services, we assume that SpiderOak maintained limited resources due to a high volume of service requests. Thus, auctioning is initiated to determine which agents shall acquire service. Due to the low price offered by AgentX ($7), it was not feasible to compete with the prices (randomly generated between $3-$20 with normal distribution) submitted by rival agents (average price of $12), thus SpiderOak was not able to accommodate AgentX's bid. Thus, our system selected the next available provider; CloudMe, where the auctioning process was repeated. As the average price submitted by rival agents was $5, AgentX's bid was considered by CloudMe for service. In case where, no perfect match existed, our framework would ask the agent to store his file locally in order not to violate his security requirements.

## V. TEST AND EVALUATION

### A. Experimental Setup

All the tests were conducted based on our engineered Cloud storage prototype system. To carry out our experiments we have used the HP ProLiant ML350p Gen8 Tower Server as our test-bed. ProLiant ML350p utilizes the Intel Xeon E5-2620 v2 CPU (6 core, 2.1 GHz, 15MB, 80W) and 128 GB of RAM. To conduct our tests we have deployed the following entities: (i) an auctioneer, (ii) interfaces to the nine selected Cloud storage providers and (iii) 50 agents, each maintaining 20 assets/files that need to be stored online (total of 1000 assets). The small number of agents and assets can be attributed to the complex and time consuming procedure of setting them up. To automate the operation of agents we have written a script that randomly generates a number of security policies and dummy text files that agents use as their bids and assets respectively.

### B. Experimental Results

*1) Dependability:* This experiment aims to demonstrate the success/failure rate in matching bids with asks. All the bids (1000 bids in total) submitted for auctioning were created based on the explicit method. All the values used for the formulation of the bids were randomly generated.

The obtained results demonstrated that 700 bids were successfully matched to Cloud storage services, where 300 left unmatched. More specifically, 250 bids were not matched due to unmet security requirements (security violations), where the remaining failed due to scarce resources. Despite the mediocre (i.e. 70%) success rate in service selection by our framework, the results are still encouraging as we have not considered any implicit bids, which would have significantly increase our success rate. Furthermore, the majority of our failed service selection attempts can be attributed to the large number of parameters comprising a bid (i.e. 16), which allow for an enormous number of possible permutations between these attributes and consequently the number of different expected bids. As there is only a small number of asks in the market and a very large number of possible bids, there is a high probability that some explicit bids will not be matched to providers.

*2) Static Security VS Adaptive Security:* Contemporary service selection frameworks are inflexible as they are not concerned with the runtime security requirements of individual assets; instead they promote "one service for all", which imposes higher costs (monetary and operational) and unmet security goals. To demonstrate the above assertion we have tested if the selection and utilization of a single Cloud storage can meet the security requirements of all of the assets maintained by an agent. More specifically, the goal of this experiment is to determine the effects of static security provision on the satisfaction of asset security requirements, in terms of insufficient or excessive security provision. If an asset retrieves insufficient security, its security is at stake, whereas if an asset

acquires excessive security it is likely that the costs (both monetary and operational) are unnecessarily high.

To perform this test we stored 20 assets in DropBox. For each asset we have randomly generated and associated a bid illustrating the asset's runtime security requirements. We have then compared the bid of each asset to the security specification (ask) of DropBox to determine if it met the exact security requirements of the assets. Based on the results we have labeled bids as: "satisfied", "excessive security" and "insufficient security". The results illustrated that the majority of the assets did not met their security goals as only six bids were satisfied, where five bids obtained unnecessary security provision and nine acquired insufficient security. Contrary, the test performed with our framework, satisfied the security requirements and cost constraints of all assets as they were treated as separate entities. We were able to observe that six bids required provision from DropBox, five bids were matched with OneDrive, three with CloudMe and the remaining six with Mega.

*3) Market VS Non-Market Service Allocation:* This experiment illustrates the limitations of non-market service allocation mechanisms and the added value of introducing market-inspired allocation methodologies when selecting and allocating services. More specifically, our experiments demonstrate the following limitations: (i) absence of monetary insensitive for both agents (cheaper prices) and SPs (higher revenues) and (ii) the over-provisioning or starvation of resource due to the underestimation or overestimation of service supply in cases of scarce resources. This rises due to the deficient prioritization of agent requests based on the criticality of their security requirements and the prices that are willing to pay to acquire services. This can enable agents that do not require the immediate utilization of resources to reserve and waste them, where refusing provision to agents in need.

To perform these tests we have used SpiderOak and 50 agents (1000 bids) in the presence of scarce resources (only 35 agents / 700 bids could be served). The bid prices submitted by the deployed agents were randomly generated with a normal distribution between \$1-\$30. To determine if the existing non-market mechanisms can prioritize and provide service to agents based on the severity of their requests we have labeled bids based on their significance: 400 bids labeled as "very significant", 150 bids as "medium significance", 350 bids as "low significance" and 100 bids as "storage for future usage". We then asserted which bids obtained service. The order of bid submission was random to simulate real-life conditions. Our results illustrated that 300 "very significant" bids, 95 bids of "medium significance", 250 "low significance" bids and 65 bids for "storage for future usage" were served. The results indicate that a big number of (i.e. 100) "very significant" bids did not receive service, where 65 bids requesting "storage for future usage" were able to acquire and waste storage resources. Contrary, the same experiment with the utilization of our reversed Posted-offer algorithm, was able to cater for 370 "very significant" bids, 130 bids of "medium significance", 180 "low significance" bids and only 20 bids for "storage for future usage". By introducing our auctioning algorithm the number of "very significant" bids drastically increased, where the number of "storage for future usage" bids radically decreased. This test demonstrated that the usage of auctioning mechanisms in the existence of scarce resources can better prioritize and more fairly serve bids based on their severity due to the absence of simplistic allocation methodologies that are founded on the "first come first served" policy used by existing service selection methodologies. Furthermore, we have observed that the revenue of SpiderOak was significantly increased in the presence of our auctioning mechanism. In the absence of the allocation mechanism SpiderOak's storage was offered for a fixed cost of \$7, witnessing a revenue of \$350 (50 agents $\times$ \$7), where in the presence of the market mechanism the profit of SpiderOak increased to \$398 due to the competition between agents for acquiring scarce storage resources. Finally, we have assessed how the profit of agents is affected in the presence of high supply and low demand cases in both market and non-market inspired environments. In the presence of a non-market allocation mechanism the cost of storage from SpiderOak remained stationary to \$7, where in the presence of our market-inspired allocation mechanism, the agents were able to obtain service for an average of \$5.3, which illustrates a significantly lowered cost for storage.

## VI. Conclusion

We have promoted and engineered an asset-driven, security-aware, service selection framework that leverages market-inspired methodologies for selecting services that best satisfy the security and cost constraints of assets. We have demonstrated its applicability on Cloud storage services and illustrated that market mechanisms are a dependable solution for securing assets. Potential future work is to determine the effects of various auction types on our framework.

## References


[1] J. Malczewski, "On the use of weighted linear combination method in GIS: common and best practice approaches", Transactions in GIS, 4(1), 2000, pp. 5-22.

[2] A.C. Squicciarini, B. Carminati and S. Karumanchi, "Privacy aware service selection of composite web services invited paper", In Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 9th International Conference Conference on, IEEE, October 2013, pp. 260-268.

[3] H. Mouratidis, S. Islam, C. Kalloniatis and S. Gritzalis, "A framework to support selection of cloud providers based on security and privacy requirements", Journal of Systems and Software, 86(9), 2013, pp. 2276-2293.

[4] E. Costante, F. Paci and N. Zannone, "Privacy-aware web service composition and ranking", In Web Services (ICWS), IEEE 20th International Conference on, June 2013, pp. 131-138.

[5] K. Shahedeh, C. Gacek, and P. Popov, "Security-aware selection of Web Services for Reliable Composition.", 2015.

[6] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future Gener. Comput. Syst. 25, June 2009, pp. 599-616.

[7] J. Ketcham, S. L. Vernon and W. W. Arlington, "A comparison of posted-offer and double-auction pricing institutions." The Review of Economic Studies 51.4, 1984, pp. 595-614.

[8] CloudMe, Available at: https://www.cloudme.com, Accessed: 11 Feb. 2016.

[9] Cubby, Available at: https://www.cubby.com, Accessed: 11 Feb. 2016.

[10] DropBox, Available at: https://www.dropbox.com, Accessed: 11 Feb. 2016.

[11] GoogleDrive, Available at: https://www.google.com/intl/en/drive, Accessed: 11 Feb. 2016.

[12] MEGA, Available at: https://mega.nz, Accessed: 11 Feb. 2016.

[13] OneDrive, Available at: https://onedrive.live.com/about/en-gb, Accessed: 11 Feb. 2016.

[14] SpiderOak, Available at: https://spideroak.com, Accessed: 11 Feb. 2016.

[15] Yandex Disk, Available at: https://disk.yandex.com, Accessed: 11 Feb. 2016.

[16] BEAR DataShare, Available at: https://beardatashare.bham.ac.uk, Accessed: 11 Feb. 2016.